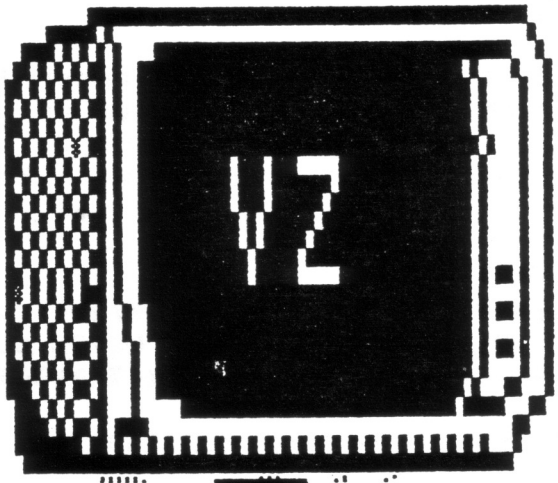
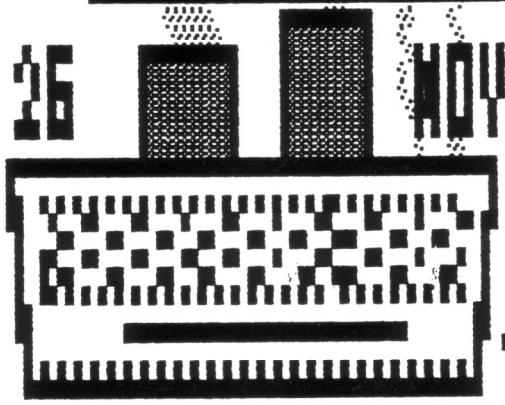
VZ-LINK MAGAZINE



ISSUE 26 NOVEMBER 88



NEW ZEALAND

VZ 300/200

COMPUTER USERS CLUB

NEWS AND VIEWS

Well,here we are with another issue of your newsletter.
I am sorry this issue is a bit late this month,but it's that time of year you know.
The little addition to the front cover is just my way of saying CONGRATULATIONS to Ross and margaret on the arrival of a boncing baby boy.
Well Done you two...

I am no longer at the same address nor have the same phone number.If you need help urgently please phone me at work.
The Phone number is 794-600 Ext 8389

Also,I am sorry that the software some people ordered is so long in arriving,but I have had problems with the master disk's.

The workshop/meeting is coming up,see details with this newsletter.

Enough of the problems etc
I hope you enjoy this issue::

****
AUCKLAND VZ 300/200 COMPUTER USERS CLUB
****

VZ USERS-WORKSHOP/MEETING

WE WILL BE HOLDING A VZ USERS WORKSHOP ON:-

SATURDAY 3 RD DECEMBER 88
STARTING AT 10-15 AM

AT THE :-

OWAIRAKA BAPTIST CHURCH
119 RICHARDSON ROAD
OWAIRAKA

BRING YOUR FAVOURITE SOFTWARE ETC
BRING YOUR PROBLEM'S

A LIGHT LUNCH WILL BE PROVIDED,BUT PLEASE BRING A PLATE TO HELP

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME AT WORK IF YOU HAVE PROBLEMS

PUT/GET ** 3 BY ROBERT QUINN

When RUN, PUTGET*3 sets up two machine code routines (PUT and GET) in a block of unused memory in the COMMUNICATIONS REGION and activates two of the old TRS80 DISK tokens which are POKEd into the BASIC subroutine (PUT/GET) located at the beginning of PUTGET*3 .

In text mode your VZ only uses the first quarter (512 bytes) of video memory to display the screen. The remaining three quarters of video memory can be used as three video stores:- VS0 (29184 to 29695), VS1 (29696 to 30207) and VS2 (30208 to 30719).

PUTGET*3 allows you to store three screens in video memory and recall them whenever you wish. The BASIC PUT/GET subroutine can be called with a GOSUB12, and offers you six options, selected by pressing the number keys 1 to 6:-

<1> GET: copies VS0 to screen
<2> GET1: copies VS1 to screen
<3> GET2: copies VS2 to screen
<4> PUT: copies screen to VS0
<5> PUT1: copies screen to VS1
<6> PUT2: copies screen to VS2

If you don't want the PUT/GET MENU to display (and so be copied to the video stores along with the rest of the screen) then use a GOSUB14 instead of a GOSUB12. The number keys will still do their jobs; you just have to remember which keys do what. Pressing any character key other than the number keys 1 to 6 will QUIT PUT/GET.

```
2 '*************************************************************
4 '*     PUTGET*3  SUBROUTINE  WRITTEN  BY  ROBERT  QUINN     *
6 '*************************************************************
8 :
10 GOTO022
12 PRINT"****GET 1 2 3"," PUT 4 5 6****"
14 K$=INKEY$:K$=INKEY$:IFK$=""THEN14ELSESOUND20,1
16 IFK$="1"THENGELSEIFK$="2"THENG1ELSEIFK$="3"THENG2
18 IFK$="4"THENPELSEIFK$="5"THENP1ELSEIFK$="6"THENP2
20 IFK$>"0"ANDK$<"7"THEN14ELSERETURN
22 POKE31556,164:POKE31567,164:POKE31579,164
24 POKE31594,165:POKE31605,165:POKE31617,165
30 FORR=31273TO31345:READB:A=A+B:POKER,B:NEXT
35 IFA<>6240THENSOUND30,2;20,1:PRINT"ERROR IN DATA":END
40 POKE31107,41:POKE31108,122:POKE31104,72:POKE31105,122
45 DATA17,0,114,183,40,19,254,58,40,15,35,254,49,17,0,116,40,7
50 DATA254,50,17,0,118,32,44,229,33,0,112,24,31
55 DATA229,33,0,114,17,0,112,183,40,21,254,58,40,17,225,35,229
60 DATA254,49,33,0,116,40,7,254,50,33,0,118,32,7,1,0,2,237,176
65 DATA225,201,225,195,151,25
```


FOR PRIVATE SALE

COMPUTER SHARER SWITCH :-
Complete ready to plug in, includes paperwork. Requires power supply or plug pack.

PRICE $50.00 & includes P/P within Aust. and is available from :-
Dave BOYCE 41 HEATHER Drive CHRISTIE DOWNS 5164 S.A.
For more info send SSAE or phone (08) 384 6574

ASSEMBLY LANGUAGE

An Overview of the VZ ROM

WRITTEN BY DANIEL AYERS
REPRODUCED WITH KIND PERMISSION
FROM CHRISTCHURCH VZ USERS GROUP

The VZ BASIC version 2.0 ROM is 16K long and occupies the bottom 16K of memory from 0000H to 3FFFH, or 0-16383 in decimal. The overall layout of the ROM is difficult to follow, and to this day I cannot see any profound pattern. When faced with the task of disassembling it by hand, there was only one way to do it. At the start we had no idea if any given byte in the ROM was code, data or junk (there is some of that). The method we used was simple. We followed the path the chip took when the computer was switched on. Execution began from byte 0, so that was where we started to disassemble from.

After many months hard work, the end result was a 700 page volume the size of a telephone book. The secrets of VZ computers are contained in this book, as are the answers to many nagging questions.

I will start with a look at some of the table of data in the ROM and explain their uses. Anyone who has ever written a program to dump the ROM to the screen will know that the familiar messages will appear (like "READY", "SYNTAX ERROR", and "BREAK"). In conjunction with a block printing routine in the ROM, your VZ uses these tables to produce those all-too-well known phrases.

The following tables are used by the block printing routine. All messages are in ASCII & are terminated with a null (00h) byte. No checking is performed, so control codes may be used.

HEX	DEC	MESSAGE
010E	00270	<Cr> VIDEO TECHNOLOGY <Cr> BASIC V2.0 <Cr> <Cr>
0380	00896	OK <Cr>
0384	00900	ERROR <Cr>
0570	01392	RUN
191E	06430	ERROR
1924	06436	<Space> IN <Space>
1929	06441	READY <CR>
1930	06448	BREAK
2178	08568	?REDO <Cr>
202E	08238	? EXTRA IGNORED <Cr>
37D0	14288	<Cr> VERIFY <Space>
3842	14402	WAITING
384A	14410	<Cr> LOADING ERROR <Cr>
385A	14426	FOUND
3860	14432	LOADING

<Cr> - Carriage Return - ASCII 13Dec 0DH
<Space> - ADCII 32Dec 20H

(Note: On the VZ, printing a Carriage Return also causes a Line Feed to be printed, moving the cursor to the next line)

The above messages (with the exception of "RUN") are printed using the block print routine at 28a7H (10407dec). Text printed by this routine should always be in NORMAL ASCII, not in inverse. To print inverted messages, the INVERSE FLAG (bit 1 of 7838H, 30776dec) should be set.

The text "RUN" starting at 570H 1392dec is never actually printed onto the screen. To economise on ROM space, the CRUN, CLOAD, & VERIFY commands use the same routine in ROM. This routine is actually the CLOAD routine by default, but by setting the verify flag (bit 3 of 7839H 30777dec) and calling the routine, a VERIFY is perfomed. The code checks this flag, and if set no data is loaded in from the file. This same routine does the CRUN function by performing a normal CLOAD and checking the AUTORUN flag (bit 6 of 7839H 30777dec). If set, "RUN" is placed into the input buffer when the load is completed. This causes the program to RUN.

There are two other main data tables in the VZ ROM. These are the KEYWORD DATA TABLE and the ERROR DATA TABLE. Both of these tables use a different method of marking the boundary between adjacent items of data than in the block printing routine.

The block printing routine uses a null byte (00H). While easy to program for, this method wastes 1 whole byte at the end of every item in the table. The Keyword and Error tables use bit 7 of the first character of each item as a marker. That is why, if you have ever done a dump of the VZ ROM, you will recognise the Keywords, but the first character of each appears garbled (probably some graphics character). If you rewrote the program to AND each byte with 127 before its ASCII value was printed on the screen, the error messages and keywords would appear correctly, but with no spaces in between. Such is life!

The KEYWORD DATA TABLE can be found in the ROM from 1650H 5712dec to 181FH 6175dec. Note that some keywords are missing (FRE, STRING$, AUTO etc.) Although the computer is capable of doing these, they have been removed from the data table (to cover up a bug in the tokenising routine - but don't tell Dick Smith that!!)

The ERROR DATA TABLE occupies ROM from 3CECH 15596dec to 3E28H 15912dec. I bet there are some errors in there you have never seen before!...

NO RESUME ERROR
RESUME WITHOUT ERROR
UNPRINTABLE ERROR
?DISK COMMAND
?SYNTAX ERROR ERROR

That last one is <u>not</u> a misprint.

VZ ROM SYSTEM VARIABLES

PROGRAM POINTERS

78A4/5	30884/5	Start of BASIC program
78A2/3	30882/3	Current line number
78DA/B	30938/9	Line number for DATA scan
78E2/3	30946/7	Current AUTO line number
78E4/5	30948/9	AUTO line increment
78EA/B	30954/5	Line where error occured
78F0/1	30960/1	Line number of error handling routine
78F5/6	30965/6	Number of last line executed
78F7/8	30967/8	Last byte executed
78F9/a	30969/70	End of BASIC program
788E/F	30862/3	Pointer to USR routine

DATA POINTERS

78A0/1	30880/1	Pointer to end of BASIC stack
78A7/8	30887/8	Keyboard buffer pointer
78B3/4	30899/900	Pointer to next string storage
78DF/E0	30943/4	Pointer to result of expression
78E8/9	30952/3	Machine stack pointer
78F3/4	30963/4	Address of decimal point during conversion
78FB/C	30971/2	Pointer to start of complex variables
78FD/E	30973/4	Pointer to start of free memory
78B1/2	30897/8	Top of memory pointer (T.O.M.)

SYSTEM DATA

7820/1	30752/3	Cursor positon (byte offset)
7838	30776	Bit 1 - Inverse Flag
7839	30777	Bit 3 - Verfiy Flag (Loading Routine) Bit 4 - Up/Down Cursor disable Bit 5 - Autorun (CRUN) flag
783B	30779	Copy of I/O latch
783C	30780	Character at cursor postion
7841	30785	Cursor flash rate
7846	30790	Bits 4,5,6 foreground colour
784C	30796	Loading message disable flag (non-zero = disable)
7899	30873	INKEY$ ASCII storage
789A	30874	ERR storage
789B	30875	Printer column
789C	30876	I/O flag - 00 = Screen 01 = Printer FF = Tape
78AA/C	30890/2	Random Number Seed
78DC	30940	FOR Active Flag
78DD	30941	Text input Flag
791B	31003	TRACE flag (non-zero = TRace ON)

SYSTEM VECTORS

These system vectors are used to alter the action of the ROM by software, without replacing any chips. The vectors are three-byte mahcine code programs that are CALLED by the ROM at certain times. They are three bytes long becase that is the minimum length required to perform a Jump (JP). Some vectors are "blocked off" by default (they contain a RET) and some branch to an error routine. Most of the unallocated tokens branch to the ?DISK COMMAND?SYNTAX ERROR routine.

VECTOR TABLE

HEX	DEC	ASSEM	COMMENT
7800	30720	JP 1C96H	Default RST 08H routine
7803	30723	JP 1D78H	Default RST 10H routine
7806	30726	JP 1C90H	Default RST 18H routine
7809	30729	JP 25D9H	Default RST 20H routine
780C	30732	RET	RST 28H default - N/A
780F	30735	RET	RST 30H defualt - N/A
7833	30771	LD A,0 RET	Printer Error Vector
787D	30845	RET	Interrupt Vector
7893	30867	IN A,0 RET	INP routine
7896	30870	OUT (0),A RET	OUT routine
79A6	31142	RET	Error routine (before message)

ENHANCING VZ BASIC CONT. . . .

Having adjusted the top of memory pointer, the address at 7804H is stored and replaced by our own. The program then locates the new routine at the top of memory. Now each time a byte is to be examined during execution it must first pass through our checkpoint. Once the origin of the call is established, the routine looks for the CLS token, 132 (84H). Only when it has been located does the routine proceed to examine the next byte. This is checked to see if it lies in the range 0-9. Once it has passed this test, the clear screen routine is implemented after first calculating the appropriate value with which to fill the screen. You will notice that not only is it necessary to check for the new command, but also to provide the routine which implements it. In this case a simple block load to the screen has been used. Control is then returned to the ROM processing routine, which prepares to examine the byte following our new command. So, as far as the VZ knows, everything is continuing normally. Tricky isn't it?

I have already successfully used this approach to produce a VZ Printer Patch, which enables all the normal printer functions for owners of EPSON or EPSON compatible printers. The COPY command is intercepted by the patch and as a result its function has been enhanced to allow a proper dump of both the LO-RES and HI-RES screens. One further enhancement that could be explored would be an extension of Basic's SOUND command. The possibilities are limited only by imagination and memory.

```
0001 ;#######################
0002 ;# ENHANCED CLS COMMAND #
0003 ;# BY LARRY TAYLOR 1986 #
0004 ;#######################
0005 ;
0006 ;THIS SECTION RELOCATES
0007 ;THE PROGRAM TO THE TOP
0008 ;OF AVAILABLE MEMORY.
0009 ;
0010 VCTR EQU  7A28H          ;SET VCTR AS 7A28H
0011      LD   SP,7700H       ;LOAD STACK POINTER
0012      LD   HL,(78B1H)     ;GET THE TOP OF MEMORY
0013      LD   BC,ENDP-NVCT   ;GET LENGTH OF PROGRAM
0014      PUSH BC             ;SAVE PROGRAM LENGTH
0015      XOR  A              ;RESET ALL FLAGS
0016      SBC  HL,BC          ;TAKE LENGTH FROM TOP OF MEMORY
0017      LD   (78B1H),HL     ;LOAD NEW TOP OF MEMORY
0018      PUSH HL             ;SAVE NEW TOP OF MEMORY
0019      XOR  A              ;RESET ALL FLAGS
0020      LD   BC,33H         ;RESERVE 50 BYTES STRING SPACE
0021      SBC  HL,BC          ;TAKE SPACE FROM TOP OF MEMORY
0022      LD   (78A0H),HL     ;LOAD START OF STRING SPACE
0023      POP  DE             ;RETRIEVE TOP OF MEMORY
0024      INC  DE             ;INCREASE BY ONE
0025      LD   HL,(7804H)     ;GET CURRENT RST10H VECTOR
0026      LD   (VCTR),HL      ;STORE IT IN 7A28H
0027      LD   (7804H),DE     ;LOAD NEW VECTOR
0028      LD   HL,NVCT        ;GET START OF PROGRAM TO MOVE
0029      POP  BC             ;RETRIEVE PROGRAM LENGTH
0030      LDIR                ;MOVE TO NEW LOCATION
0031      CALL 1B4DH          ;DO A NEW
0032      JP   1A19H          ;JUMP TO READY MESSAGE
```

```
0033 ;
0034 ;START OF THE PROCESSING
0035 ;ROUTINE FOR NEW COMMAND.
0036 ;
0037 NVCT EXX                    ;SAVE ALL REGISTERS
0038      LD   HL,1D5BH          ;CHECK TO
0039      POP  DE                ;SEE IF THE
0040      OR   A                 ;RETURN
0041      SBC  HL,DE             ;ADDRESS
0042      PUSH DE                ;IS 1D5BH
0043      EXX                    ;RESTORE ALL REGISTERS
0044      JP   NZ,1D78H          ;IF NOT GO TO NORMAL PROCESSING
0045      PUSH HL                ;SAVE STRING ADDRESS
0046      CALL 1D78H             ;GET NEXT VALUE FROM STRING
0047      JR   NZ,CONT           ;IF NOT ZERO THEN CONTINUE
0048 POP  POP  HL                ;ELSE RESTORE STRING ADDRESS
0049      LD   DE,(VCTR)         ;RETRIEVE ORIGINAL VECTOR
0050      PUSH DE                ;AND JUMP
0051      RET                    ;TO IT
0052 CONT CP   84H               ;CHECK FOR CLS TOKEN
0053      JR   NZ,POP            ;IF NOT FOUND RETURN TO CALLER
0054      INC  HL                ;MOVE TO NEXT VALUE IN STRING
0055      LD   A,(HL)            ;GET NEXT VALUE AFTER CLS TOKEN
0056      SUB  30H               ;REDUCE IT TO RANGE 0-8
0057      JR   Z,EXEC            ;IF ZERO THEN EXECUTE COMMAND
0058      LD   B,8               ;LOAD B REG WITH UPPER LIMIT
0059 CMPR CP   B                 ;CHECK IF A=B
0060      JR   Z,EXEC            ;IF YES THEN EXECUTE COMMAND
0061      DJNZ CMPR              ;REDUCE B AND CONTINUE CHECK
0062      JR   POP               ;NO MATCH SO RETURN TO CALLER
0063 EXEC POP  DE                ;RETRIEVE OLD STRING ADDRESS
0064      POP  DE                ;RETRIEVE OLD RETURN ADDRESS
0065      LD   DE,1D1EH          ;LOAD NEW RETURN ADDRESS
0066      PUSH DE                ;SAVE NEW RETURN ADDRESS
0067      INC  HL                ;MOVE TO NEXT VALUE IN STRING
0068      PUSH HL                ;SAVE CURRENT STRING ADDRESS
0069      ADD  A,A               ;MULTIPLY CLS
0070      ADD  A,A               ;VALUE BY 16 TO
0071      ADD  A,A               ;CALCULATE THE
0072      ADD  A,A               ;COLOUR OFFSET
0073      JR   NZ,SKIP           ;IF RESULT NOT ZERO THEN SKIP
0074      INC  A                 ;IF ZERO INCREASE TO ONE
0075 SKIP ADD  A,7FH             ;ADD 127 TO GET GRAPHICS BLOCK
0076 ;
0077 ;CLEAR SCREEN ROUTINE
0078 ;
0079      LD   HL,7000H          ;LOAD START OF SCREEN ADDRESS
0080      LD   (7820H),HL        ;SET CURSOR POSITION
0081      LD   DE,7001H          ;LOAD START OF SCREEN PLUS ONE
0082      LD   BC,01FFH          ;NUMBER OF BYTES TO MOVE
0083      LD   (HL),A            ;LOAD GRAPHICS BLOCK INTO HL
0084      LDIR                   ;DO A BLOCK FILL OF THE SCREEN
0085      POP  HL                ;RETRIEVE STRING ADDRESS
0086      RET                    ;RETURN TO 1D1EH TO CONTINUE
0087 ENDP DEFB 0                 ;END OF PROGRAM MARKER
```

BLOCK TRANSFERS BY C. HOBROUGH

There are many situations in machine language programming where you need to move the contents of a whole block of memory to another location without processing it in any way. Data files, screen images and simple prompt messages all usually consist of a string of bytes which are stored in one place and then moved somewhere else to be used or displayed.

Ultimately, each byte must be moved individually but the Z-80 processor has four instructions which help speed things up. These are :-

LDI - LoaD and Increment
LDIR - LoaD and Increment then Repeat
LDD - LoaD and Decrement
LDDR - LoaD and Decrement then Repeat

Their use is best illustrated by the following example :-

```
LD HL, FROM     ;source
LD DE, TO       ;destination
LD BC, COUNT    ;counter
LDIR            ;automated transfer
```

This represents block transfer at it's simplest. First the pointers are set up, with HL containing the address of the first byte to be moved and DE containing the address of it's destination. Then BC is loaded with the number of bytes in the block. The last instruction then carries out the whole transfer. It moves the first byte then increments both HL and DE (HL=HL+1, DE=DE+1) to point to the next byte and decrements the count in BC (BC=BC-1). This is repeated until BC equals zero, thus moving the whole block. As you can see, it's a very powerful instruction.

An example of this in use is the scrolling of the screen which takes place as you list a program. Each time the bottom line is completed the whole screen must be moved up one line to make room for more at the bottom. To rewrite the entire screen, in its new position from the program table in memory would be a waste of time because the program is not stored in the form that appears on the screen and must be processed. It's much quicker to take the block of Video RAM which starts at the first byte of the second line and move it back 32 bytes to the start of the first line. The bottom line is then cleared and the next line of code processed and written in. As you can see, the whole screen is shifted in much less time than it takes to write the new bottom line.

Suppose you have a program which does something similar but you want to scroll the other way. If you used the routine above then FROM would be the start of the first line and TO would be the start of the second line. The result would be that the second line would be overwritten before it was moved and you'd end up with the first line repeated 15 times. The answer is to replace LDIR with LDDR and start at the bottom of the screen, at the end of the next to last line. This instruction, after moving each byte, decrements HL and DE as well as BC and so moves backwards through the block. If the size of the block is less than the distance being moved then it doesn't matter which instruction is used but if not then use LDIR to move down in memory and LDDR to move up.

Sometimes the exact length of the block is not known in advance. Records in a data file may be terminated with a carriage return as a marker and your block transfer routine will have to test for this marker to see if the end has been reached. In this case something like the following will be used :-

```
     LD  HL, FROM      ;set pointers
     LD  DE, TO        ;
     LD  BC, COUNT     ; & counter
     LD  A, 0DH        ; CR code for comparison
NEXT CP  (HL)          ; compare character with CR
     JR  Z, END        ; exit if found
     LDI               ; else transfer byte and increment pointers
     JP  PE, NEXT      ; return for next byte if BC <> 0
END  RET               ; exit
```

The pointers and counter are set up as before and the code of the marker is placed in A. The current byte is then compared with the marker to see if the end has been reached, in which case the transfer terminates. If there is no match then LDI moves the byte, increments HL and DE, and decrements BC as before but does not automatically repeat. If BC has not reached zero the PARITY/OVERFLOW flag will be set and the routine will jump back for the next byte. If not then it is cleared and the routine ends. The counter and the test for zero are optional. If there is no maximum length to be tested then BC need not be set (it will still be decremented) and line 8 could be JR NEXT.

There are two ways in which these routines can be modified to make them more general. Firstly, the pointers and counter may not be known ahead of time but be calculated by the main program and held as variables. This makes the IMMEDIATE addressing mode impossible and the ABSOLUTE or EXTENDED mode must be used. Secondly, the routine might have to work out the direction of transfer in order to use the appropriate instruction, as mentioned before. The most obvious way to do that is to subtract TO from FROM and test for the sign of the result. Fortunately the HL register pair may be used as a 16 bit accumulater and there is a 16 bit SuBtract with Carry instruction available. The general routine might end up looking like this:-

```
   FROM  DEFW nn        ; pointers and counter
   TO    DEFW nn        ; variables defined in
   COUNT DEFW nn        ;  main program
;
TRNSF LD   HL, (FROM)   ; load pointers
      LD   DE, (TO)     ;
      LD   BC, (COUNT)  ; & counter
      SCF               ; set carry flag
      CCF               ; complement (ie. clear) carry flag
      SBC  HL, DE       ; FROM - TO
      LD   HL, (FROM)   ; reload FROM
      JP   M, MVUP      ; jump if result negative
      LDIR              ; else transfer down
      JR   END          ; & exit
MVUP  LDDR              ; transfer up
END   RET               ; exit
```

As you can see the pointers are now passed to the routine from outside and both forms of the transfer instruction are used. SBC is a subtract with carry so the carry flag must be cleared first. FROM has been corrupted by the subtraction and must be reloaded but, as load instructions do not affect the flags, the SIGN flag is still valid.

A similar routine could be written using the LDI and LDD instructions in a more general way with tests for markers or address locations as needed. There are many possible modifications but the essential technique is defined by the nature of the block transfer instructions themselves. Obviously the same effect could be achieved by a combination of load, increment and decrement instructions but the result would be more lines of code and slower results.

NB. :- The transfer process is in fact a copy and the original data is unchanged until it is subsequently overwritten.

.... VZ SOFTWARE LIST ....

This is an updated list. It is compiled by Mr Eddie Tomes.

* COMMERCIAL ** SOFTWARE *
For The
Dick Smith Electronics
VZ 200/300 Colour
COMPUTER

Program Name	Program Distributor
EDUCATIONAL PROGRAMS	
Elementry Geometry	DSE
Flashcard 1 & 2	DSE
Flashcard 3 & 4	DSE
Maths Armada	DSE
Metric Spycatcher	DSE
Music Writer	DSE
Speed Reading	DSE
Statistics I	DSE
Statistics II	DSE
Spellomatic 1 & 2	DSE
Spellomatic 3 & 4	DSE
Typing Teacher	DSE
Whizkid Spycatch	DSE
Word Matching	DSE
Speed & Accuracy	LYS
Study Aid	LYS
Block Puzzler	VSW
Capping	VSW
Coordinates	VSW
Edudisk	VSW
European Capitals	VSW
Keyboard	VSW
Maths Countdown	VSW
Maths	VSW
Meatpies	VSW
Microscope	VSW
Name the Towns	VSW
Plus & Minus	VSW
Queensland	VSW
Tower Of Hanoi	VSW
Wordmatching	VSW
Radioman Pack	HIR
Parent/Teacher Pack	HIR
Morse Code	MIL
Code Maker/Break	McC
Maths Pack	McC
Number Challenge	McC
Word Challenge	McC
Block Puzzler	LAS
Coordinate	LAS
Edudisk	LAS
Keyboard	LAS
Maths Countdown	LAS
Meatpies	LAS
Microscope	LAS
Name The Towns	LAS
Smartle	LAS
Tower Of Hanoi	LAS
Type Teacher	BNE
GAMES PROGRAMS	
Snake Maze	LOW
Air Traffic Controller	DSE
Asteroids	DSE
Attack Killer Tomato	DSE
Backgammon	DSE
Biorythm	DSE
Blackjack	DSE
Checkers	DSE
Chess	DSE
Circus	DSE
Crash	DSE
Dam Patrol	DSE
Defence Penetrator	DSE
Diamond Poker Machine	DSE
Digout	DSE
Dracula's Castle	DSE
Duel	DSE
Dynasty Derby	DSE
Formula One	DSE
Galaxon	DSE
Ghost Hunter	DSE
Hamburger Sam	DSE
Hangman	DSE
Hoppy	DSE
Horse/Rabbit Racing	DSE
Invaders	DSE
Kamakazi	DSE
Knights & Dragons	DSE
Ladder Challenge	DSE
Learjet	DSE
Lunar Lander (Laser)	DSE
Lunar Lander (VZ300)	DSE
Matchbox	DSE
Maze Of Argon	DSE
Missile Attack	DSE
Othello	DSE
Penguin	DSE
Pharoah's Curse	DSE
Planet Patrol	DSE
Poker	DSE
Slot Mach/Russian Rou	DSE
Space Ram	DSE
Star Blaster	DSE
Super Snake	DSE
Tennis	DSE
Tennis / Golf Lesson	DSE
Tote Racing	DSE
VZ-Panic	DSE
Adventure Island	LYS
Alien Escape	LYS
Alien Rain	LYS
Asteroid Field	LYS
Beat The Bug	LYS
Castle	LYS
Caverns Of Venus	LYS
City Blitz	LYS
Convoy	LYS
Cub Scout	LYS
Enchanted Princess	LYS
Goldaine Adventure	LYS
Graphic Golf	LYS
Haunted House	LYS
Land Of Scrod	LYS
Little Packer	LYS
Lost Sword	LYS
Melbourne Cup	LYS
Mousetrap	LYS
Pharoah's Tomb	LYS
Pyramid Adventure	LYS
Russian Roulette	LYS
Slot Machine	LYS
Ten Pin Bowling	LYS
Triffids : 2040 AD	LYS
Two Up	LYS
Ufo's	LYS
VZ Star Trek	LYS
VZ Star Trek II	LYS
Yahtzee	LYS
Boskone Alert	COS
Defence Penetrator	COS
Morgoth	COS
Rally Racer	COS
Road Warrior	COS
Air Traffic Controller	VSW
Blackjack	VSW
Escape River	VSW
Factory	VSW
Flight Deck	VSW
Flying Saucer	VSW
Haunted Mansion	VSW
Learjet	VSW
Mansion / Nova	VSW
Monopoly VZ	VSW
Poker Machine	VSW
Scotland Yard	VSW
Trivial Cult	VSW
Word Squares	VSW
Biorythm	MIL
Pak Man	MIL
Super Invaders	MIL
Airlocks Maze	CEL
Battle Galaxy	CEL
Battle Of Vietnam	CEL
Cataclysm Catapillar	CEL
Conan The Barbarian	CEL
Galactic Fighter	CEL
Ghoulbusters	CEL
Jailhouse Breakout	CEL
Lunar Landers 2020	CEL
Minatour Perilless	CEL
Reactor Ddestruction	CEL
Royal Poker	CEL
Thermonuclear War	CEL
Titanium Triology	CEL
3D Merciless Maze	CEL
Golf	CAR
Skiing	CAR
Star Pilot	CAR
Target	CAR
Astro Save	TAI
Bright Idea	TAI
Cavern Chaos	TAI
Circuitry	TAI
Cross Over	TAI
Enervac	TAI
Flag Climber	TAI
Fighter	TAI
Fuse	TAI
Ghostly Maze	TAI
Grave Digger	TAI
Hunter	TAI
Missile Safe	TAI
Mower Mania	TAI
Pattern Guess	TAI
Phantom Driver	TAI
Piano Songs	TAI
Save Me	TAI
Shark Bait	TAI
Shoot	TAI
Snake	TAI
Snake Trails	TAI
Snake Trap	TAI
Three Hundred	TAI
Word Teaser	TAI
Enchanted Princess	McC
Hi - Jinks	McC
Gamblers Pack	McC
Graphic Golf	McC
Sultans Palace	McC
Ten Pin Bowling	McC
VZ Racer	McC
Star Treker	APE
The Lost Crown	APE
Blackjack Royale	VDU
Castle Greystone	VDU
FBI 2001	VDU
Galctic Empires	VDU
Haunted Mansion	VDU
Knights Quest	VDU
Scotland Yard	VDU
Snake Duel	VDU
Stellar Wars	VDU
Trivial Cult	VDU
VZ Monopoly	VDU
Air Traffic Controller	LAS
Bust Out	LAS
Cosmic Rescue	LAS
Escape River	LAS
Keno	LAS
Learjet	LAS
Light Pen Puzzle	LAS
Magnum Quest	LAS
Solo Battleships	LAS
3D Maze	LAS
VZ Wars	LAS
Teacher Zapper	SOR
Dracula's Maze	BNE
Octapus	BNE
Base Defender	LYO
Memory	LYO
To The Hills	LYO
Graphics 1	LYO
Graphics 2	LYO
Krichon	LYO
Rally	LYO
Draw	LYO
Venture	LYO
Air Traffic Controller	BJE
Asteroids	BJE
Blackjack	BJE
Checkers	BJE
Circus	BJE
Defence Penetrator	BJE
Galazian Adventure	BJE
Ghost Hunter	BJE
Golf Lesson	BJE
Hangman	BJE
Hoppy	BJE
Invaders	BJE
Knights & Dragons	BJE
Ladder Challenge	BJE
Learjet	BJE
Panic	BJE
Planet Patrol	BJE
Poker	BJE
Rally X	BJE
Star Blaster	BJE
Super Snake	BJE
Tennis	BJE
BUSINESS PROGRAMS	
Database	DSE
Discounted Cashflow	DSE
Financial Cashflow	DSE
Mailing List	DSE
Matrix	DSE
Portfolio Management	DSE
Word Processor	DSE
Card File	LYS
Check Record	LYS
Decision Evalulator	LYS
Lemonade Stand	LYS
Master Finance	LYS
Metric Conversions	LYS
Racing Selector	LYS
Cash Book Ledger	VSW
Cheque Ledger	VSW
Database VZ	VSW
Le'VZ D'base	VSW
Le'VZ Statement	VSW
Mailing List	VSW
Quickwrite	VSW
Laser Calc	LAS
Mark 64 Database	LAS
Disk Word Processor	LAS
Address Book	DAW
Data Master	DAW
Word Processor	DAW
Electronic Aid	APE
Finance Helper	APE
Matrix Utilities	APE

Super Word Processor	BRO
VZ Database	VDU
P.I.N. NBR. File	BNE

****

UTILITY PROGRAMS

****

Disassembler	DSE
Editor Assembler	DSE
Hex Utilities	DSE
Intro. To Basic	DSE
Sprite Generator	DSE
VZ-Kon 80	LYS
Hackers Delight	CEL
Program Copier	CEL
Screen Drawer	CEL
Array / Restore	VSW
Cmerge/Delete/Renum	VSW
Colour Graphics	VSW
Copy / Protect	VSW
Disassembler	VSW
Misc Guard	VSW
Disk Copy	VSW
Disk Ops 1	VSW
Disk Ops 2	VSW
Editor Assembler	VSW
Extended Basic (W.O.)	VSW
File Search	VSW
Load XX80 Files	VSW
Monitor / Debugger	VSW
Protect	VSW
Search Tape	VSW
Utilitys	VSW
VZ Epson Printer Patch	VSW
VZ Extended Basic(S.O.)	VSW
Weaving Drafts	VSW
VZ Debug	MIL
Graphics Pack	MIL
Art Gallery	LAS
Colour Hardcopy	LAS
Copy/protect	LAS
Diskguard	LAS
Disk Tool Kit	LAS
Disassembler	LAS
DSE Word Pro Patch	LAS
Editor/Assembler disk	LAS
Epson Printer Patch	LAS
Extended Basic (G.W.)	LAS
Fast Disk & Menu	LAS
Filesearch	LAS
Magic Paint	LAS
VZ Basic Compiler	LAS
VZ Convert	LAS
Colour Graphics	PAM
Monitor / Debugger	WOB
Renumber / Merge	WOB
VZ Extra Commands	LEH
Disk Menu	BNE
Poster Shop	BNE
Art Gallery	BNE
Tape I.D.	BNE
Tape / Disk List	BNE
Program Copier	SAR
Retrieval Program	BRO
Copier	SUN
VZ Chip 8	GRI
Graf Star	VDU
Name Star	VDU
Shop Star	VDU
Screen Ed	MTS
E+F WP Disk Patch	HVG

****

SOFTWARE DISTRIBUTORS

DECODING

****

APE Ape Software
BJE S.Bjelic
BNE Brisbane VZ Workshop
BRO G.Browell
CAR Adam Carter
CEL Celestron
COS Cosmic
DAW L.Dawson
DSE Dick Smith Electronic
GRI Griffin
HIR J.Hirst
HVG Hunter Valley VZ Grp
LAS Laserlink
LEH G.Lehman
LOW B.Low
LYO R.Lyons
LYS Lysco
McC G.McCleary
MIL R.S.Miller
MTS M. Taylor Software
PAM Pam Software
RHO C.Rhodes
SAR N.Sarafoudis
SOR J.Sortland
SUN E.Sundstrup
TAI L.Tait
VDU VZ Down Under
VSW Vsoftwarez
WOB W.Obrist

****

A number of programs appear under several suppliers and are sold on behalf of the one writer over a period of years. Also several different versions may have been available as the programs are improved as time goes by.

****

Some Software companies have either closed or are now distributing under new names.

****

LYSCO ....Stopped selling VZ software 31JUL 86 but no further word on new distributor for these programs.

CELESTRON ....Also N. Sarafoudis . These programs very hard to obtain even with a personal approach .

COSMIC .....Appears to have been taken over by DSE.

R.S.MILLER ....Closed down .

J.HIRST .... Also traded as HI-COM software.

BJELIC ... Most of this software appears to be pirate copies.

VDU is closing down it's Software company although the User Group will still keep operating.Although the postal address has changed.

****

** CORRECTION .....**

VSW program listed as FLIGHTDECK appears to be incorrect name for Air Traffic Controller.

****

PLEASE NOTE ##############

As Compiler of this list I do not hold copies of , Or have distribution rights for all of these programs but will assist in contacting the applicable Software . House if required. The addresses are not included in this list as a number have changed or are no longer trading and I wish to prevent as many problems as possible for users in obtaining the programs of their choice by giving the latest information known.

****

To avoid all the mentioned software being lost to the future VZ user it would be an advantage to have a Central Library to hold a copy of these programs so that they would be available when present support is no longer available. This would be done only once copyright restrictions have passed or writers permission has been obtained . Appropiate charges would then be applied to cover expenses of tapes or disks and postage etc.

Please advise me of your thoughts on this .

****

I would be interested to hear of any new or old software that is not listed so this catalogue can be updated at regular intervals . Please advise program name and also name and address of writer / distributor to enable correct records to be kept.

********** Compiled origionally :-

July 1986.

Updated :-

December 1986.

June 1987.

November 1987.

September 1988.

****

By:-.
Eddie Tomes
3 Kilkenny St.
Capalaba
Qld. 4157 .
Telephone 07 - 3902797 H

****

SOFTWARE FOR SALE

We now have the following software available for sale.
All prices shown include postage and packing anywhere in New Zealand.
Please make sure that the memory of your VZ is greater than or equal to the memory required by the software that you wish to purchase.
Memory sizes indicated do not include video ram.

ALL PRICES ARE SHOWN IN NEW ZEALAND DOLLARS.

BLACKJACK ROYALE	16K	$15.00	HAUNTED MANSION	22K	$15.00
CASTLE GREYSTONE	32K	$15.00	KNIGHTS QUEST	32K	$18.00
DATA	16K	$18.00	NAMESTAR/SHOPSTAR	22K	$15.00
FBI-2001	31K	$18.00	SCOTLAND YARD	16K	$ 8.00
GALACTIC EMPIRES	16K	$15.00	TRIVIAL CULT	16K	$ 8.00
GRAPHSTAR	22K	$24.00	VZ MONOPOLY V2.0	22K	$18.00

PLEASE MAKE ALL CHEQUES AND MONEY ORDERS PAYABLE TO-

AUCKLAND VZ 300/200 COMPUTER CLUB.

VZ DISASSEMBLER FOR SALE

This Disassembler is written by Peter Hickman.
And is also distributed as the "Laserlink Disassembler".

Peter is offering this very fine program to user group members for NZ$24.00.
All orders can be placed through this club,as peter has offered a cheaper price for bulk purchases.

The program is available on tape or disk.And comes in a 4 program version and
You choose which section you use depending on what memory you have.

ALL THESE PROGRAMS CAN BE DEMONSTRATED AT THE WORKSHOP IF REQUESTED